Московский Государственный Университет им. М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Суперкомпьютеров и Квантовой Информатики

Отчёт

Параллельная реализация решения СЛАУ с помощью метода отражений.

Работу выполнил:
Лесцов Б.А.
423 группа

2018

Постановка задачи и формат данных.

Задача: Реализовать параллельный алгоритм решения СЛАУ с помощью метода отражений.

Постановка задачи:

Заданы невырожденная вещественная матрица *A* размера *NxN* и вещественный вектор *b* длины N такие что система *Ax=b* имеет единственное решение. Необходимо найти вектор *x,* являющийся решением этой системы.

Компиляция:

1) С помощью gnu make:

> make

2) С помощью cmake:

> mkdir build

> cd build

> cmake ..

Запуск:

1) Считать матрицу из файла:

> mpirun -np ‘число процессов’ build/bin/Prak ‘путь к файлу с матрицей’

Пример*:*

> mpirun -np 2 build/bin/Prak data/mat.txt

2) Сгенерировать с помощью функции f(i, j), заданной в source/main.cpp:

> mpirun -np ‘число процессов’ build/bin/Prak ‘N’ ‘N+1’, где N – размер матрицы системы

Пример*:*

> mpirun -np 4 build/bin/Prak 1024 1025

Формат входных файлов:

Входной файл:

1) Файл с матрицей размера *Nx(N+1)* в текстовом виде. В начале файла располагаются два числа m, n типа size_t – размеры матрицы. Далее следуют n*m вещественных чисел – сама матрица. Последний столбец этой матрицы – это вектор *b.*

Формат входных файлов:
В стандартный поток вывода будет выдано решение заданной СЛАУ в формате:

x_1 = ‘значение переменной x_1’
...
x_i = ‘значение переменной x_i’
...
x_N = ‘значение переменной x_N’

Далее следует строка:

Mat_size ‘размер матрицы’ Comm_size ‘число процессов’
Forward_Time_(microsec) ‘время приведения к верхнетреугольному виду в микросекундах’
Backward_Time_(microsec) ‘время обратного хода метода Гаусса’
diff ‘невязка’

Описание алгоритма:
Метод отражений основан на разложении матрицы A системы $Ax=b$ в произведение унитарной матрицы на верхнюю треугольную. Матрица A называется унитарной, если она удовлетворяет уравнению $A \cdot A^* = E$, где A^* - матрица, сопряженная с A. Вещественные унитарные матрицы называются ортогональными.

По своей структуре метод отражений близок к методу Гаусса, но исключение проводится с помощью матриц отражения, которые являются унитарными и эрмитовыми. Достоинством метода отражений является единая схема вычислительного процесса, не зависящая от структуры матрицы.

Теорема. Пусть S и l произвольные вектор-столбцы, причем вектор l имеет единичную длину. Тогда найдется такой вектор W, что построенная по нему матрица отражения $U = E - 2ww^*$ переведет вектор S в вектор, коллинеарный вектору l, т.е. $Us = \alpha l$.

Вектор W строится по правилу $w = \frac{1}{\rho}(s - \alpha l)$, где $|\alpha| = \sqrt{(\alpha,\alpha)}$, $\arg\alpha = \arg(s,l) - \pi$, $\rho = \sqrt{(s - \alpha l, s - \alpha l)} = \sqrt{2|\alpha|^2 + 2|\alpha|\,|(s,l)|}$.

Будем преобразовывать расширенную матрицу систему по правилу

$A_{k+1} = U_{k+1} A_k$, $k = 0,1,\ldots,n-2$

с помощью умножения слева на последовательность матриц отражения $U_1, U_2, \ldots, U_{n-1}$. Для построения матрицы U_1 на первом шаге метода в качестве вектора S берется первый столбец расширенной матрицы, а в качестве вектора I - координатный вектор $I = (1,0,0,\ldots,0)^T$. В силу выбора векторов S и I все координаты первого столбца расширенной матрицы, кроме первой, после выполнения первого шага метода будут равны нулю.

Пусть уже построена матрица A_k, у которой $a_{i,j}^{(k)} = 0$, $i > j$, $j = \overline{1,k}$. Теперь в качестве S и I берутся вектора

$$s = (0,\ldots,0,a_{k+1,k+1}^{(k)}, a_{k+2,k+1}^{(k)},\ldots,a_{n,k+1}^{(k)})^T,\ I = (0,\ldots,0,1,0,\ldots,0)^T,$$

где в векторе I единица стоит на $k+1$-ом месте. После выполнения k-го шага метода отражений получим матрицу A_{k+1}, у которой все элементы, стоящие ниже главной диагонали, в первых $k+1$-ом столбцах будут равны нулю. Невозможность выполнения очередного шага связана только с равенством нулю вектора S, а это невозможно, так как матрица A является невырожденной.

После $(n-1)$-шага получим матрицу, первые n столбцов которой образуют верхнюю треугольную матрицу L. Система уравнений, соответствующая полученной расширенной матрице, равносильна исходной системе. Значения неизвестных находятся аналогично обратному ходу метода Гаусса

$$x_n = -\frac{a_{n,n+1}^{(n-1)}}{a_{n,n}^{(n-1)}},\ x_i = -\frac{a_{i,n+1}^{(n-1)} + \sum_{j=i+1}^{n} a_{i,j}^{(n-1)} x_j}{a_{n,n}(n-1)}, i = n-1, n-2, \ldots, 1$$

Параллельная версия алгоритма подразумевает разделение матрицы A между процессами по столбцам. При этом на каждом этапе работы алгоритма один из процессов подсчитывает вектор *w,* и рассылает его остальным процессам. Получив нужный вектор, каждый процесс производит обновление всех столбцов своей части матрицы по правилу: *y_i -= 2*alpha*w, где alpha -* скалярное произведение w и y_i .

Во время параллельного выполнения обратного хода метода Гаусса процессы последовательно вычисляют переменные x_i, после чего отсылают остальным процессам вычеселнный x_y, а также вектор $x_i * y_i$

где y_i – i-й столбец верхнетреугольной матрицы.

Результаты выполнения.

Тестирование производилось на системе Blue Gene/P. Использовались матрицы размеров 1024 x 1024, 2048 x 2048, 4096 x 4096 и 8192x8192. Для 8192x8192 приведены результаты для 128, 256, 512 и 1024 процессов, так как на меньшем количестве процессов программа работает слишком долго.

Результаты:

Число процессов	Прямой ход	Обратный ход	Общее время	Невязка
Размер матрицы: 1024x1024				
1	6.6995	0.013508	6.713008	3.57882e-05
2	3.338913	0.021545	3.360458	1.29046e-05
4	1.739243	0.02804	1.767283	3.84221e-05
8	0.890013	0.026201	0.916214	4.42852e-05
16	0.484198	0.025044	0.509242	3.61034e-05
32	0.268348	0.02396	0.292308	3.00219e-05
64	0.163723	0.02356	0.187283	2.17098e-05
128	0.10904	0.02338	0.13242	0.00106634
256	0.080012	0.023066	0.103078	2.70167e-05
512	0.064396	0.023097	0.087493	0.000144786
1024	0.058591	0.024329	0.08292	2.64742e-05
Размер матрицы: 2048x2048				
1	58.50323	0.05235600	58.555586	0.000126021
2	30.277927	0.07043600	30.348363	7.90297e-05
4	15.675619	0.07618900	15.751808	6.29209e-05
8	7.636962	0.08679100	7.723753	6.37356e-05
16	4.139699	0.08396400	4.223663	2.32327e-05
32	2.05058	0.07846300	2.129043	9.03271e-05
64	1.137106	0.07603600	1.213142	0.000628378
128	0.682852	0.07483800	0.75769	0.000114271
256	0.440021	0.07376300	0.513784	0.000113385
512	0.324681	0.07361100	0.398292	5.58626e-05
1024	0.267481	0.07630000	0.343781	0.00114095

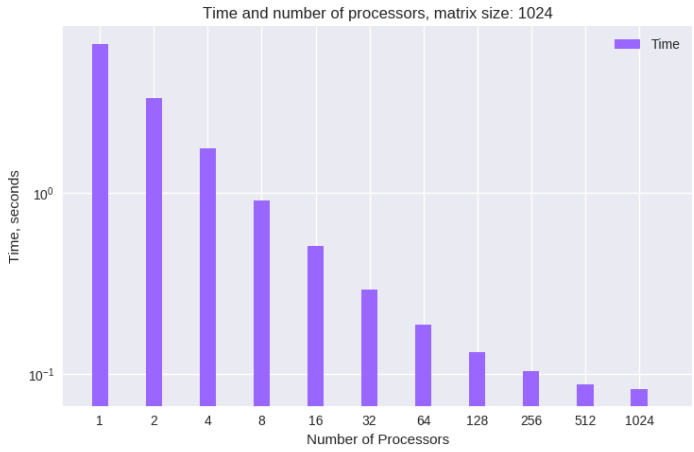
Число процессов	Прямой ход	Обратный ход	Общее время	Невязка
Размер матрицы: 4096x4096				
1	530.27333	0.21608200	530.48942	0.000230225
2	265.54816	0.25276500	265.80093	0.000292556
4	159.58657	0.21237900	159.79895	0.00051608
8	80.731031	0.26185300	80.992884	5.48158e-05
16	40.607092	0.27973300	40.886825	0.000493042
32	20.89823	0.25430700	21.152537	9.00789e-05
64	10.678252	0.24271000	10.920962	0.00123484
128	5.863177	0.23589500	6.099072	0.000822494
256	3.23064	0.23252800	3.463168	0.00015566
512	2.152498	0.23179400	2.384292	0.000101221
1024	1.467595	0.23715100	1.704746	0.000281008
Размер матрицы: 8192x8192				
128	43.587618	0.7798990	44.367517	0.0005900
256	24.133031	0.7698590	24.90289	0.0001109
512	13.552089	0.7611100	14.313199	0.0005624
1024	8.270344	0.7739660	9.04431	0.0002126

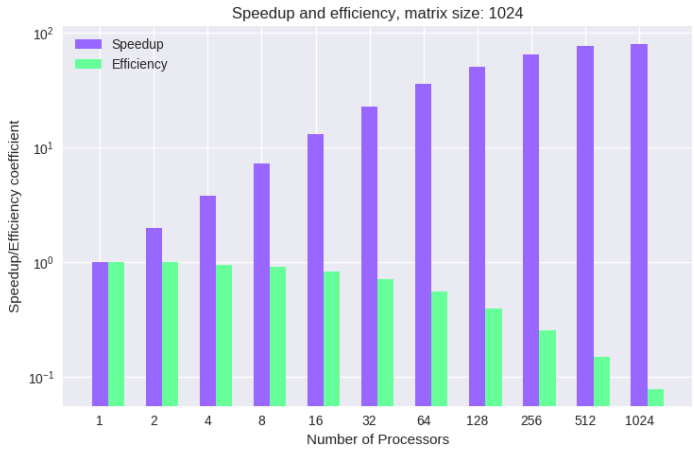
Ускорение и эффективность:

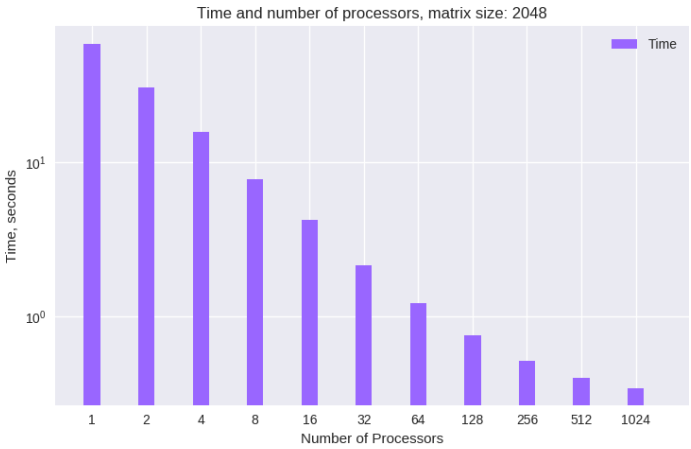
Число процессов	Ускорение	Эффективность	Ускорение	Эффективность
	1024x1024		**2048x2048**	
1	1.000	1.000	1.0	1.000000
2	1.9976467	0.998823	1.92945	0.964724
4	3.7984907	0.949622	3.71739	0.929347
8	7.3268996	0.915862	7.58123	0.947654
16	13.182353	0.823897	13.8637	0.866481
32	22.96553	0.717672	27.5032	0.859476
64	35.844193	0.560065	48.2677	0.754183
128	50.69482	0.396053	77.2817	0.603763
256	65.125517	0.254396	113.969	0.445192
512	76.72623	0.149855	147.017	0.287142
1024	80.957646	0.079060	170.328	0.166336
	4096x4096		**8192x8192**	
1	1.0	1.000000	-	-
2	1.99581	0.997907	-	-
4	3.31973	0.829933	-	-
8	6.54983	0.818728	-	-
16	12.9746	0.810911	-	-
32	25.0792	0.783726	-	-
64	48.5753	0.758990	-	-
128	86.9787	0.679521	-	-
256	153.18	0.598361	-	-
512	222.493	0.434558	-	-
1024	311.184	0.303890	-	-

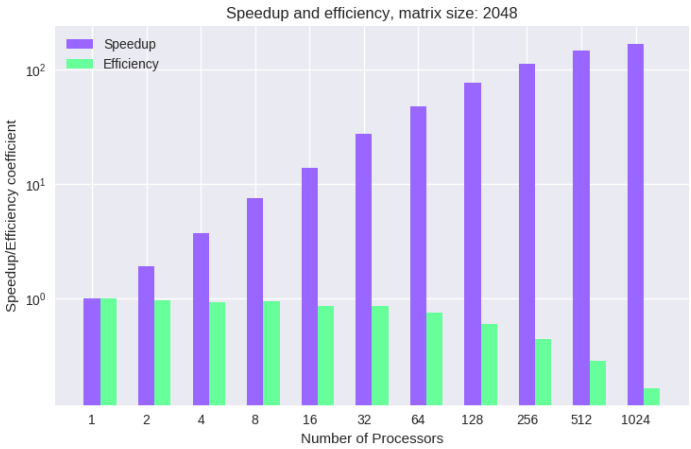
Графики

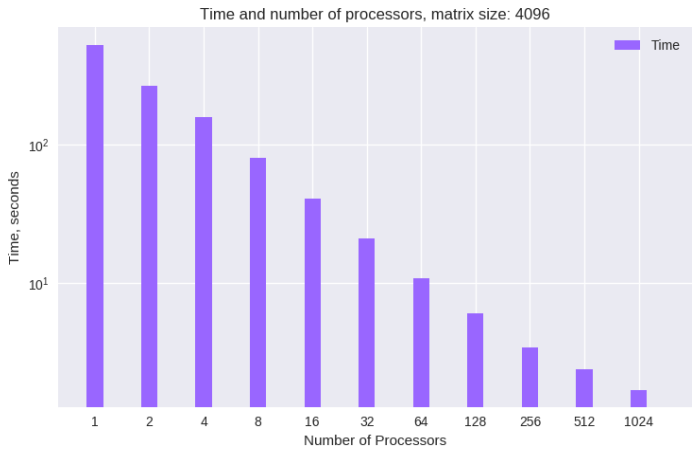
Далее приедены графики для матриц 1024x1024, 2048x2048, 4096x4096 и 8192x8192.

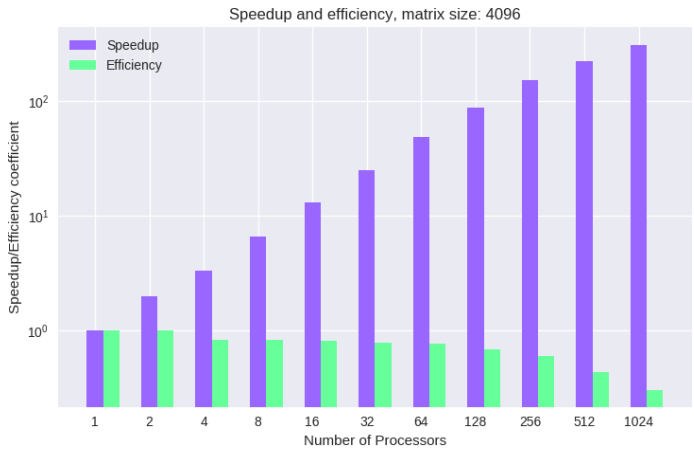


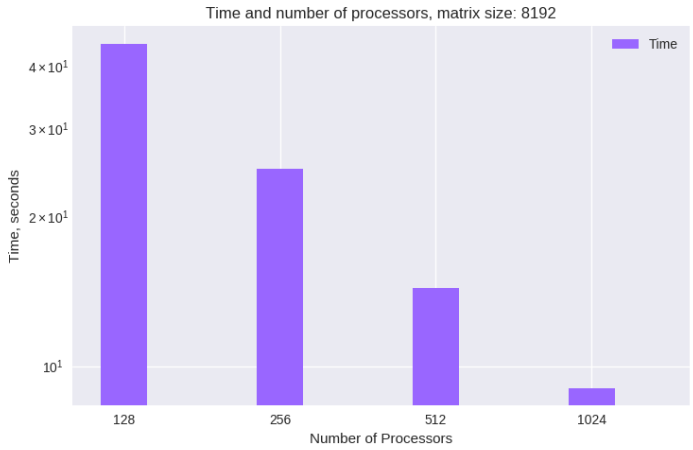


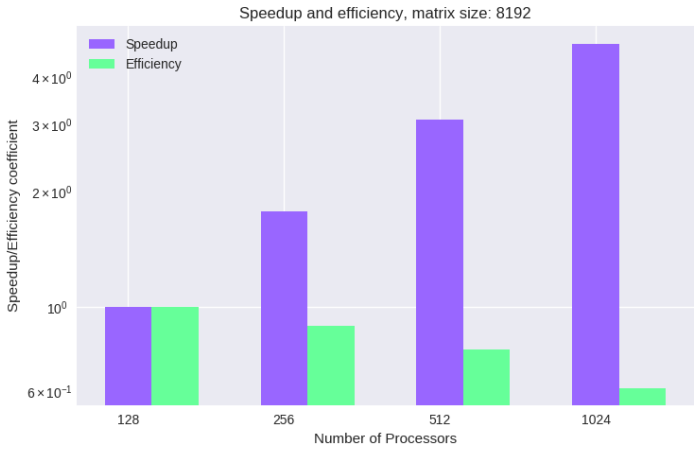












Основные выводы

С увеличением числа процессов время выполнения значительно уменьшается. Задача решения СЛАУ с помощью метода отражений эффективно распараллеливается, и можно получить значительное ускорение при достаточно высокой эффективности.